應用以樣版爲基礎之軟體框架提昇軟體開發品質

張志宏、盧志偉、林志豪、楊明峰

摘 要

近年來，由於軟體產業的發達，全世界及台灣的軟體系統的需求量愈來愈大，企業對於軟體工程的重視及需求愈來愈大，但是開發者對於軟體工程的導入卻經常反而無所適從，一方面是軟體工程導入不易，開發人員教育訓練時間過長；一方面所使用的軟體輔助開發工具相互整合來源資料亦有難處，彼此間無法協調整合。如此多樣的不確定因素讓開發者之發展環境持續在改變而讓開發整體更加困難，導致業界對於系統開發太過於客製化，無法充分發揮軟體再使用的特性，進而降低系統開發成本，而且軟體開發方法、軟體開發流程與軟體輔助開發工具無法相互整合，以至於不能發揮整體效益，亦是另一個讓整體開發及維護更加的複雜而不容易掌控的因素。

本篇研究的目的在於探討如何導入以軟體樣版爲基礎之軟體框架，應用於產學合作之企業資源規劃系統及供應鏈系統的發展，期望透過軟體樣版將問題領域特性與商業邏輯分離，簡化軟體開發過程時系統開發人員對於商業邏輯應對時因實務經驗不足，導致需求分析與系統實作無法充分被管控，產生開發品質與時程難以掌控的問題。進而提昇軟體品質，降低軟體開發的成本，同時提高軟體的可維護性。

關鍵詞：設計樣版、軟體發展流程、軟體框架、XML。

張志宏：修平技術學院資訊管理系助理教授
盧志偉：修平技術學院資訊管理系副教授
林志豪：修平技術學院電子計算機中心技士
楊明峰：修平技術學院電子計算機中心技士
投稿日期：970826　接受刊登日期：971014

K Applying Pattern-based Software Framework to Improve the Quality of Software Development

Chih-Hung Chang, Chih-Wei Lu, Chih-Hao Lin, Ming-Feng Yang

Abstract

In recent years, development of the software industry and demand for software systems have increased rapidly, but developers often does not know whose suggestion to follow regarding methodologies of software engineering. One reason for that is the difficulty in applying new software engineering technologies. Developers take a long time to train. Another reason is the difficulty in integrating CASE toolsets. So many indeterminate factors make the development process more and more complex. On the other hand, software development is too customized, and software reuse is difficult. The reasons above are the cause for software development and maintenance to become more complex and difficult to control.

In this paper we survey the popular of an open source software pattern-based framework, and the development of an ERP/support chain system. Based on software patterns, developers can separate development and business so as to reduce problems caused by the developer's lack of business experience. The quality of the product can thus be enhanced, software development costs be reduced, and software maintenance be improved.

Keywords: Design Pattern, Software Development Process, Framework, XML.

Chih-Hung Chang, Assistant Professor, Department of Information Management, HIT.
Chih-Wei Lu, Associate Professor, Department of Industrial Management, HIT.
Chih-Hao Lin, Software Engineer, Computer Center, HIT.
Ming-Feng Yang, Software Engineer, Computer Center, HIT.

壹、簡介

在使用物件導向技術時，物件導向中的繼承機制可以使得軟體元件可以再使用，而軟體再使用的目的是試著將現存的軟體元件盡可能的再利用，這種方法明顯的可以降低軟體開發的成本，因此產生高可再用的軟體元件是軟體工程的一個主要目的，然而當一般程式設計師著重於程式碼再利用時，對於設計概念的再使用往往會被忽略，而設計樣版(Design Patterns)對於設計結構提供了一個清楚的概念。設計樣版描述了系統構成元件間繼承(Inheritance)及參考(Reference)關係。設計樣版的主要目的之一便是幫助軟體工程師瞭解物件在特殊應用領域中的共有特徵。

所謂設計樣版可以說是用來解決一些常見程式撰寫問題的一種程式寫作模式，這種寫作模式幫助我們在撰寫某些特定的程式或解決特定的問題時，可以很快的按照一定的模式及步驟來撰寫程式碼，這種方法使得我們的程式架構不論是在解讀上或是未來的維護上都有更好的表現，設計樣版除了幫助我們在傳統的程式撰寫外，更讓一個系統的規格架構更清楚明白，並增加系統的維護性。

近年來由於 WWW 的發展，加上 Java[14]技術的成熟，許多應用程式的發展也紛紛走向網頁應用程式(Web Application)的架構，許多軟體技術也紛紛應用到網頁應用程式上，如設計樣版(Design Patterns)、軟體框架(Framework)等等，例如 Apache Struts[12]、Spring Framework[13]都是是開放源碼(Open Source)框架，其主要是爲了解決企業應用程式開發複雜性而產生的。框架的主要優勢之一就是其分層架構，分層架構允許開發者選擇使用哪一個組件，同時爲 J2EE 應用程式開發提供整合的框架。像這些新興的網頁軟體技術，便有助於網頁應用程式的開發，但這些相關技術的整合，卻仍缺乏一個有系統的整理，因此本研究便希望運用這些開放源碼的軟體技術，發展一個可以應用於網頁應用程式之軟體框架，一方面解決目前在網頁應用系統中程式缺乏一個良好架構的問題，並透過導入這些開放源碼技術，降低企業開發成本，同時也提供企業在導入、應用這些開放源碼一個經驗。

本篇論文的架構如下，首先我們將在第二章說明相關的背景及研究，在第三章我們將討論這些開放源碼的軟體技術及討論我們如何利用框架整合這些開放源碼技術來開發大型系統，緊接著在第四章我們將我們的開發經驗做一個案例研究，最後是我們對於這個議題的結論。

貳、相關研究

一、設計樣版(Design Patterns)

自 Gamma et al [1]在 1994 年提出設計樣版(Design Patterns)之後，設計樣版的應

用常被定位在解讀性高，快速建置軟體架構上，在 Gamma et al 列舉出了三大類設計模版，這些設計模版都可以應用在像是資料結構、事件觸發處理、訊息的傳遞等等常見的設計，加強了系統在設計時的可讀性及可再利用性。Alexander 等人在其研究中 [19] 解釋，所謂的樣板 (Pattern) 是 "a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice"，由其原文可知設計樣版主要目的是幫助在特殊應用領域中瞭解軟體元件中共有的架構與設計概念格式的基礎。希望經常發生的問題能透過以往成熟的經驗建立一套解法，無需每次重新構想設計。

設計樣板[1] [2]成功地將這些在軟體設計過程中時常見的問題，透過整合設計實務及專家的經驗成爲一組具備更好特性的可再用的軟體元件，設計樣板具備前瞻的現代軟體分析與設計的方法[18]，可以幫助設計師容易的在使用具備眾所週知且成功的專家設計概念及架構，它也幫忙程式設計師在設計軟體系統時，利用再使用元件建立系統及可再用提高系統的可再用性的另一個技術方案。

二、框架(Framework)：

在進行軟體重整工程中，如果我們用 Framework-based 來發展系統在對於一個特定的問題領域來說是具有相當高的再使用性[3]。框架(Framework) [4] [5] [6]是一組共同相互合作完成同一個目的的抽象及實體類別所組成，其中包含了相關的介面及其定義[20]，這些類別是由一組相同領域的可再用的元件所構成。設計者可以透過繼承這些類別去產生符合其需求的功能。基本上框架是提供一個環境來支援軟體開發的所有活動。但是我們在做 Framework 的結合時有時候會發生一些問題，所以在 Mattsson 和 Bosch 提出 Framework 結合時所遇到的問題[7]，和爲什麼會發生這些問題的原因，然後最後有提出使用現有的 design patterns 來幫忙解決問題。雖然 framework 來幫忙我們做軟體重整工作時，已經可以有比較高的效率，但 Framework 是針對於一個特定的問題領域，而 Design Pattern 是使用於一般性的問題領域，只要找出的這些 Design Patterns 是符合我們的需求的，就可以拿到任何一個問題領域裡來使用，這又比 Framework 還更具有普遍性，也更具有更高的重使用性。而在 Jacobsen 等人所提出的 Paper 裡 [8]，也有提到在發展一個 Framework 系統時，分別在分析、設計和實作這三個部份如果都利用 Pattern 來幫助我們發展系統的話，那我們在發展系統時就依據 Design Pattern 的特性來一個階段一個階段的發展下去，也比較少花時間來做上個階段到下個階段對照的工作，而是有一貫性的作業

下去。所以說我們提出以 Design Pattern 爲基礎的來做軟體重整工程，相信可以更快速的來做軟體重整工程，並且以後在做維護工作時也能更容易。

近年來由於網頁應用程式的蓬勃發展，一些可以應用在網頁程式上的軟體框架也紛紛被提出，如 Spring Frameworkv[13]是一個開放源碼框架，是爲了解決企業應用程式開發複雜性而產生的軟體框架。

三、XML

XML[9] 是由 World Wide Web Consortium (W3C)所制定的標準，一種延伸式的標記語言，具有擴展性(Extensibility)、結構性(Structure)、描述性(Description)、確認性(Validation)等特性。同時 XML 具有跨平台的功能，對於不同的作業系統、硬體設備、應用軟體、多元的輸入模式，開發者可以自行制定符合己身需求的標記(Tag)，做結構性的描述，促使相同的一份文件呈現不同的規格，適用於不同的軟體，符合不同的設備、滿足多重的輸入方式。

在不同領域資料的描述，XML 也具貢獻，ebXML(Eectronic Business)描述企業資料規格[10]、VoiceXML 描述聲音的標準規格[9] [11]、MathML 描述數學領域的標準規格[9]、…等等。這些標準的描述爲因應二十一世紀電子商務時代，所有資料都希望以網路的方式傳輸，提供快速、精確、便利的個人化服務。

XML 目前也應用在許多開放源碼系統的環境設定上，如 Apache 及本研究所用到的 Spring framework 控制層敘述等。

參、系統架構

本研究中使用 Apache Struts MVC[12] 、 Spring Framework[13] 與 Hibernate[15]等網頁應用程式之開放源碼技術，其主要的概念分述如後；接著，我們將描述我們如何整合這些技術來設計軟體系統。

一、Struts

Struts 是 Apache 軟體基金會下 Jakarta 項目的一部分。Struts 框架的主要架構設計和開發者是 Craig R.McClanahan。Struts 是目前 Java Web MVC 框架中相當受重視的技術之一，最早是在 Smalltalk 中出現。經過長達五年的發展，Struts 已經逐漸成長爲一個穩定、成熟的框架。其特色在於將模型與介面相互分離，程式碼即可實現可管理性與可重用性，而其中 Model 通常也被稱爲「交易邏輯」。

但是 Struts 某些技術特性上已經落後於新興的 MVC 框架。面對 Spring Framework MVC、Webwork2 這些設計更精密，擴展性更強的框架，Struts 受到了前所未有的挑戰。但站在產品開發的角度而言，Struts 仍然是最穩妥的選擇。

Struts 有一組相互合作的類別（元件）、Servlet 以及 Jsp Tag Lib 組成。基於

Struts 構架的 Web 應用程式基本上符合 JSP Model2 的設計標準，可以說是 MVC 設計模式的一種變化類型。根據上面對 framework 的描述，我們很容易理解爲什麼說 Struts 是一個 Web Framework，而不僅僅是一些標記庫的組合。但 Struts 也包含了豐富的標記庫和獨立於該框架工作的實用程式類別。Struts 有其自己的控制單元（Controller），同時整合了其他的一些技術去實現模型層（Model）和觀點層（View）。在模型層，Struts 可以很容易的與資料庫連接存取技術相結合，包括 EJB、JDBC 和 Object Relation Bridge。在觀點層，Struts 能夠與 JSP、 Velocity Templates、XSL 等等這些表示層元件相結合。MVC（Model-View-Controller）[23]的模式如圖 1 所示。

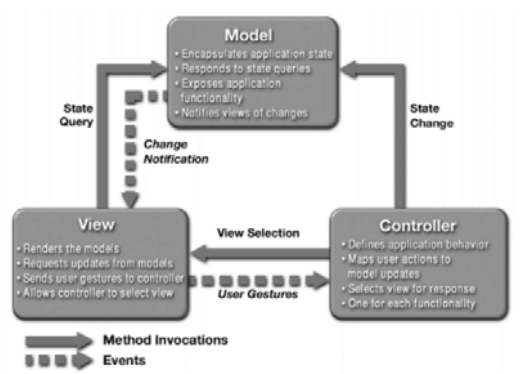

圖 1. MVC 模式

二、Spring 框架[13]

Spring 實際上是 Expert One-on-One J2EE Design and Development [21]一書中所闡述的設計思想的具體實現。在 One-on-One 一書中，Rod Johnson 倡導 J2EE 實用主義的設計思想，並隨書提供了一個初步的開發框架實現（Interface21 開發包）。而 Spring 正是這一思想的更全面和具體的實現。Rod Johnson 在 Interface21[22]開發包的基礎之上，進行了進一步的改造和擴充，使其發展爲一個更加開放、清晰、全面、高效的開發框架。

Spring 是一個開放源碼框架，它是爲了解決企業應用開發的複雜性而產生的。Spring 使用基本的 JavaBeans 來完成以前只可能由 EJB 完成的事情變得可能了。然而，Spring 的用途不僅限於伺服器端的開發。從簡單性、可測試性和疏鬆耦合的角度而言，任何 Java 應用都可以從 Spring 中受益。Spring 框架是一個分層架構，由 7 個定義良好的模組組成。Spring 模組構建在核心容器之上，核心容器定義了創建、配置和管理 Bean 的模式，如圖 2 所示。

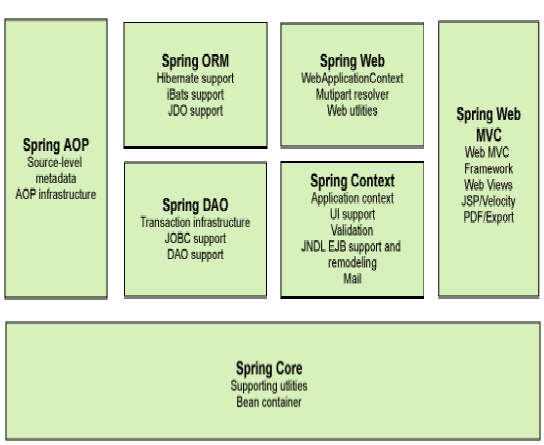

圖 2. Spring framework 架構

組成 Spring 框架的每個模組（或元件）都可以單獨存在，或者與其他一個或多個模組聯合實現。每個模組的功能如下：

- ●核心容器：核心容器提供 Spring 框架的基本功能。核心容器的主要組件是 BeanFactory，它是工廠模式的實現。BeanFactory 使用控制反轉（IoC）模式將應用程式的配置和倚賴性規範與實際的應用程式碼分開。
- ●Spring Context：Spring Context 是一個配置檔案，向 Spring 框架提供 Context 訊息。Spring Context 包括企業服務，例如 JNDI、EJB、電子郵件、國際化、校驗和調度功能。
- ●Spring AOP：透過配置管理特性，Spring AOP 模組直接將物件導向程式功能整合到了 Spring 框架中。所以，可以很容易地使 Spring 框架管理的任何物件支援 AOP。Spring AOP 模組爲基於 Spring 的應用程式中的物件提供了事務管理服務。透過使用 Spring AOP，不用倚賴 EJB 組件，就可以將宣告性事務管理整合到應用程式中。
- ●Spring DAO：JDBC DAO 抽象層提供了有意義的錯誤處理層次架構，可用該架構來管理錯誤處理和不同資料庫供應商拋出的錯誤訊息。錯誤處理層次架構簡化了錯誤處理，並且大幅地降低了需要設計的錯誤處理原始碼數量（例如打開和關閉連接）。Spring DAO 的 JDBC 導向的錯誤處理遵從通用的 DAO 錯誤處理層次架構。
- ●Spring ORM：Spring 框架插入了若干個 ORM 框架，從而提供了 ORM 的物件關係工具，其中包括 JDO、Hibernate 和 iBatis SQL Map。所有這些都遵從 Spring 的通用事務和 DAO 錯誤處理層次架構。
- ●Spring Web 模組：Web Context 模組建立在應用程式 Context 模組之上，爲基於 Web 的應用程式提供了 Context。所以，Spring 框架支援與 Jakarta Struts 的整合。Web 模組還簡化了處理多部分請求以及將請求參數繫結到對象領域的工作。
- ●Spring Web MVC 框架：MVC 框架是一個全功能的構建 Web 應用程式的 MVC 實作。透過策略界面，MVC 框架變成爲高度可配置的，MVC 容納了大量觀點層技術，其中包括 JSP、Velocity、Tiles、Itext 和 POI。

三、Hibernate

Hibernate 是「物件/關係對應」（Object/Relational Mapping）的解決方案，簡寫爲 ORM，簡單的說就是將 Java 中的物件與物件關係，映射至關聯式資料庫中的表格與表格之間的關係，Hibernate 提供了這個過程中自動對應轉換的方案。

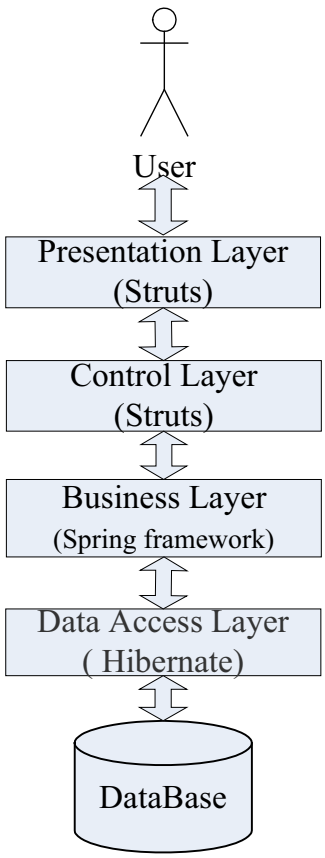

圖 3. System Architecture

在本研究中我們規劃了如圖 3 的系統架構，藉以整合以上三種軟體技術。

其中 Struts 將負責與使用者溝通之 Presentation Layer 以及 Control Layer 等模組的設計規範，而有關於 Business Layer 將透過 Sprint Framework 來達成，至於 Data Access Layer 則是透過 Hibernate 技術與資料庫溝通。各階層間的訊息交換與物件對應關係，則是利用 XML 來完成。Struts、Sprint 與 Hibernate 三者間的整合。

對於整個整合完成之後的架構模型，如圖 4 所示：

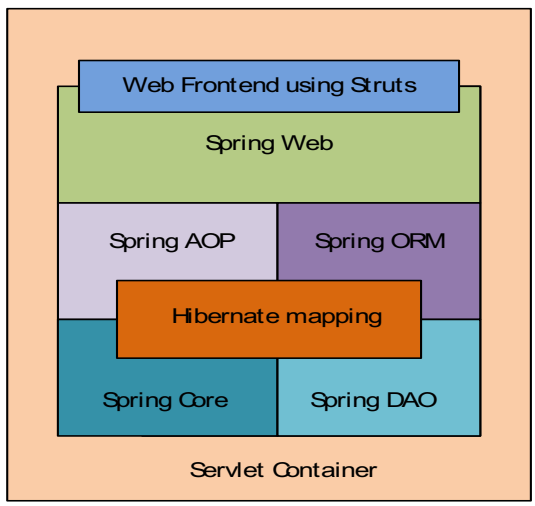

圖 4. System Integration Model

爲了順利整合各項開放源碼工具，使用 Spring Framework 爲整合架構的核心基礎，其中包含了之前所提到的 AOP、ORM、DAO 與 Core 作爲提供建置基礎架構的類別庫，橋接前端網頁 Struts 與後端 Hibernate 二者之間的交易資料通道，此架構模型建構在一個 Servlet 的容器當中，最常見的有 Apache Tomcat[16]，或其他知名軟體供應商的 Servlet Container。

在本研究中，我們將上述技術整合成以下架構，方便原始碼的管理與分類，如圖 5 所示：

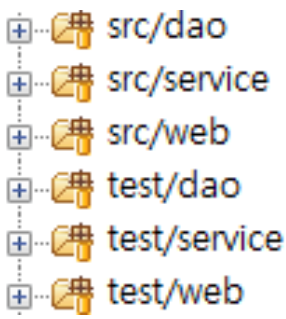

圖 5. Package Tree

支援以 Struts 爲架構的所有與使用者互動及資料輸入(Presentation Layer)，我們稱爲 Action Bean 皆放置於 src/web 統一維護。支援與 Spring 協同整合進而產生核心商業邏輯(Business Layer)則置於 Src/Service。而以 Hibernate 支援將資料庫實體物件化(Data Access Layer)的實作則置於 Src/Dao。所有物件在被發展的同時，也同時會產生一組相對應的測試案例(Test Case)，這對於提昇軟體可靠度與軟體品質監管極爲有效，使用的工具爲 JUnit API[17]。

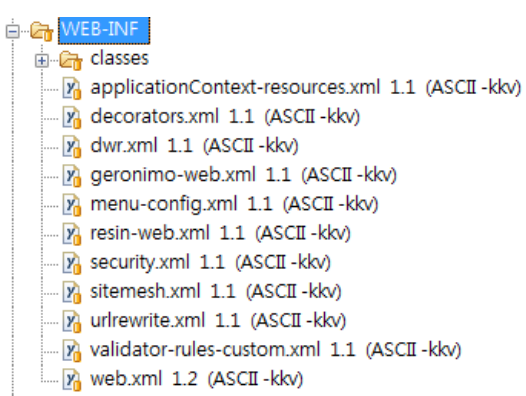

圖 6. Spring Control File

相較於程式碼套件設計規劃扮演更爲重要的則是 Spring 的控制層(Control Layer)，控制層中所有設定檔皆以 XML 來描述，如圖 6 所示：

以 applicationContext-resources.xml 爲例，如圖 7 所示：

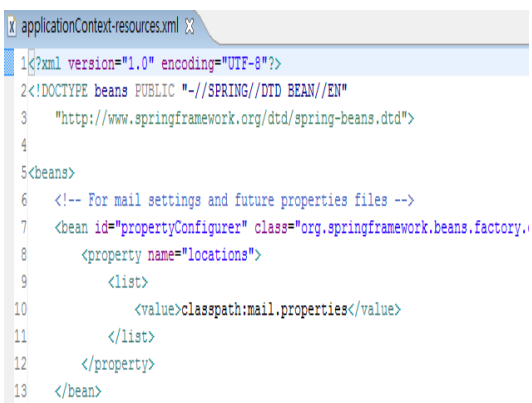

圖 7. applicationContext-resources.xml 內容

Spring 的框架中提供了一個 BeanFactoryPostProcessor 的實作類別：org.springframework.beans.factory.config.PropertyPlaceholderConfigurer。藉由這個類別，可以將一些組態設定，移出至.properties 檔案中，如此的安排可以讓 XML 定義檔負責系統相關設定，而.properties 檔可以作爲客戶根據需求，自定義一些相關的參數，例如載入 mail.properties，而此.properties 檔詳細載明郵件伺服器的相關資訊，如圖 8 所示。

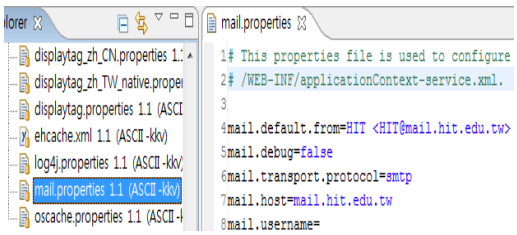

圖 8. mail.properties 內容

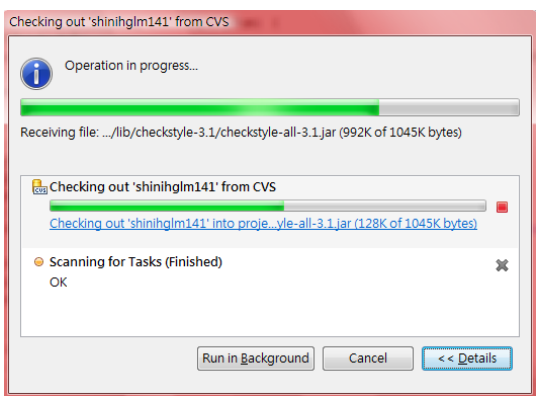

圖 9. Project Check-out using CSV

由於一般的大型系統開發，需要團隊合作，爲了提昇團隊軟體開發效率與程式各版本演進管制，導入使用團隊協同作業工具 CSV(Concurrent Versions System) ，如圖 9 所示。

由於相關的開放源碼技術差異相當大，所以在整個研究中，我們使用同樣是開放源碼工具的 *appfuse* 來做爲整合工具。在本研究中，我們主要採用 Eclipse 作爲系統整合開發環境。

肆、案例研究

爲了要驗證這個架構的有效性及可使用性，我們以兩個網頁應用程式開發專案案例作爲驗證的對象，一爲書籍流通 ERP 系統，另外一個爲跨國傳統產業 GLM 系統。其中在 ERP 這個專案中，我們一共用了 5 位有經驗的程式設計師在兩個不同的地區協同開發，專案內容包含 33 個資料表格及 28,700 行程式碼。另一個 GLM 專案中，總共使用 6 名有經驗的程式設計師在兩個不同的地區協同開發，總專案規模爲將近 100 個資料表及 86,500 行程式碼。

由於系統架構龐大，圖 10 及圖 11 是這個 ERP 這個專案中的 Form view 及 Action view 套件圖簡圖。

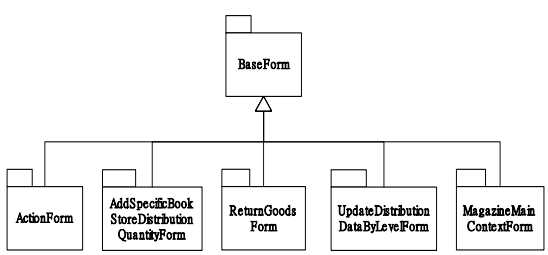

圖 10. MVC Form view of ERP project

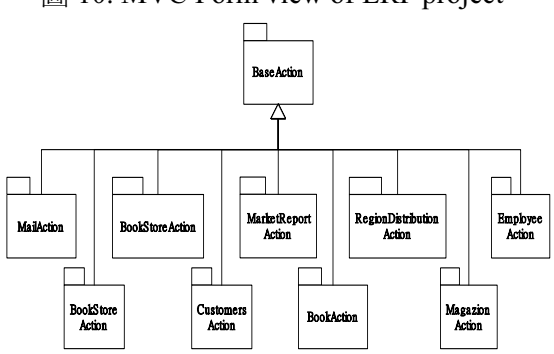

圖 11. MVC Action view of ERP project

表 1 是這兩個案例系統的開發時間。表 2 則是我們對於使用 Struts 爲主樣板網頁應用程式開發技術及使用傳統 JSP 開發方式的比較。

表 1. 系統開發時間

	ERP	GLM
開發時間	43 天	57 天

表 2. Struts 及 JSP 技術比較

	Struts	JSP
協同開發	較佳	較弱
維護性	較佳	較弱
再使用性	較佳	較弱
學習時間	慢	快

表 3 則是使用 Hibernate3 及 JDBC3.0 來做爲資料庫連接存取技術的比較。

表 3. Hibernate 3 及 JDBC 3.0 比較表

	Hibernate	JDBC
資源需求	大	小
效能	快	慢
開發時間	快	慢
學習時間	慢	快

最後我們將在第五章說明我們的結論及心得。

伍、結論

依循上述研究方法與實作流程，我們嘗試使用 Spring Framework 整合框架，並將實作 MVC 的部份使用 Struts，對於資料表的操作則使用 Hibernate，而整體軟體發展架構控制則交給 Spring Core 類別，將可真正落實軟體的分層負責，善加使用其他發展工具，強化軟體可靠度與簡化協同工作的複雜度，將有助於降低網頁應用程式軟體開發成本，提昇軟體品質。同時我們也應用了本次的研究心得與成果，應用於出版流通及傳統產業跨國 GLM 系統開發，成功透過框架及樣版技術，解決網頁應用程式結合舊有主從系統之系統整合問題，同時我們也將本次的成果導入本校財產管理系統開發案中，成功整合舊有財產資料庫至網頁應用程式，延續系統生命週期。

雖然根據我們的實務經驗發現使用 Spring Framework、Struts 及 Hibernate 可以大幅提高系統的可維護性即可再用性，但由於使用 Spring Framework、Struts 及 Hibernate 需要許多前置作業及良好的系統規劃，因此若是開發大型網頁應用程式透過本次研究成果，可有效將系統模組化、節省維護成本，若是小型網頁應用程式，則不建議採用。

參考文獻

[1]E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, pp. 12 - 37, Reading Mass.: Addison-Wesley, 1994.

[2]R.E. Johnson and B. Foote, “Designing Reusable Class,” *Journal of Object-Oriented Programming*, Vol. 1, No. 2, pp. 22-35, June/July 1988.

[3]S. Moser and O. Nierstrasz, O. “The Effect of Object-Oriented Frameworks on Developer Productivity,” *IEEE Computer,* Vol. 29, No.9, pp. 45-51, 1996.

[4]B.H.L. Betlem, R.M. van Aggele, J. Bosch, and J.E. Rijnsdorp, *An Object-Oriented Framework for Process Operation*, Technical report, Department of Chemical Technology, University of Twente, 1995.

[5]M. Mattsson, *Object-Oriented Frameworks*, Licentiate Thesis, Department of Computer Science, Lund University, 1996.

[6]S. Sparks, K. Benner, and C. Faris, “Managing Object-Oriented Framework Reuse,” *IEEE Computer*, Vol. 29, Issus 9, pp. 52-61, September 1996.

[7]M. Mattsson and J. Bosch, “Framework Composition: Problems, Causes and Solutions,” the *Proceedings Technology of Object-Oriented Languages and Systems,* pp. 203-214, 1997.

[8]E. E. Jacobsen, B. B. Kristensen, and P. Nowack, “Patterns in the Analysis, Design and Implementation of Frameworks,” The *Proceedings of Proceedings of the 21st International Computer Software and Applications Conference*, page 344, 1997.

[9]World Wide Web Consortium, http://www.w3.org, accessed 2006/12/1.

[10]R. J. Glushko, J. M. Tenenbaum, and B. Meltzer, “An XML Framework for Agent-based E-commerce”, *Communications of The ACM*, Vol. 42, No.3, pp. 106-114, March 1999.

[11]J. Ding, Y. Huang, and C. W. Chu, “Video Database Techniques and Video-on-Demand”, *Handbook of Distributed Multimedia Databases: Techniques and Application*, the Idea Group Publishing, USA, 2001.

[12]*Apache Struts*, The Apache Software Foundation, http://Struts.apache.org/1.3.9/userGuide/index.html, accessed 2007/ 7/15.

[13]R. Johnson et. al., *The Spring Framework - Reference Documentation*, Interface21, http://www.springframework.org/docs/reference/index.html, accessed 2007/4/30.

[14]*The Source for Java Developers*, Sun Developer Network, http://java.sun.com, accessed 2007/8/12.

[15]*Hibernate*, Red Hat Middleware, http://www.hibernate.org/, accessed

2007/4/12.

[16]*Apache Tomcat*, The Apache Software Foundation, http://tomcat.apache.org/, accessed 2007/5/12.

[17]K. Beck, E. Gamma, and D. Saff, *JUnit 4.1*, http://www.junit.org/index.htm, accessed 2007/3/12.

[18]C. Larman, *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design,* Englewood Cliffs NJ: Prentice-Hall International, 1997.

[19]C. Alexander, S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiksdahl-King, and S.A. Angel, *A Pattern Language*. Oxford University Press, New York, 1977.

[20]D.J. Chen and T.K. Chen, "An Experimental Study of Using Reusable Software Design Frameworks to Achieve Software Reuse," *Journal of Object-Oriented Programming*, Vol. 7 No.2, pp. 56-67, May 1994.

[21]R. Johnson, *Expert One-on-One J2EE Design and Development*, Wrox Press, 2002.

[22]R. Johnson, *Introduction to the Spring Framework*, May 2005, http://www.theserverside.com/tt/articles/article.tss?l=SpringFramework, accessed 2007/10/2.

[23]Model-View-Controller, Java BluePrints, http://java.sun.com/blueprints/patterns/MVC-detailed.html